UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/729,015	12/04/2000	Minoru Takimoto	FUJR 18.034	6757

7590 03/08/2006

KATTEN MUCHIN ZAVIS ROSENMAN
575 MADISON AVE
NEW YORK, NY 10022-2585

EXAMINER	
ROCHE, TRENTON J	

ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 03/08/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/729,015	TAKIMOTO, MINORU	
	Examiner	Art Unit	
	Trenton J. Roche	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 December 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-8,10-18 and 20-28* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-8,10-18 and 20-28* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *04 December 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to communications filed 5 December 2005.

2. Per Applicant's request, amended claims 1, 10, 11, 20 and 21 have been entered. Claims 1-8, 10-18 and 20-28 are pending.

Response to Arguments

3. Applicant's arguments filed 5 December 2005 have been fully considered but they are not persuasive.

Per claims 1, 8, 10, 11, 18, 20, 21 and 28:

The Applicant states that neither Apfel et al. nor Kullick et al. teach or reasonably suggest version information being defined in different namespaces, when written in programming language C++, for preventing a collision of names of functions and variables. In response, it is noted that the newly added claim limitation is not positively recited in the claim, and furthermore, the scope of the claim cannot be currently ascertained, as there is no indication of what must be written in C++ for the newly added limitations to occur. As such, because the limitation regarding version information being defined in different namespaces only occurs in relation to "something" being written in C++, that "something" being indefinite and undefined, the claim is hereby being interpreted as not requiring version information to be defined in different namespaces, since the claim is not requiring anything to be written in C++. As such, the prior rejection is proper and maintained.

Per claims 2-7, 12-17 and 22-27:

The Applicant states that claims 2-7, 12-17 and 22-27 are allowable as being dependent on an allowable base claim. As was shown above, the rejections regarding independent claims 1, 8, 10, 11, 18, 20, 21 and 28 are proper, and as such, the argument that claims 2-7, 12-17 and 22-27 are allowable as being dependent on an allowable base claim is considered moot. The rejections regarding claims 2-7, 12-17 and 22-27 are proper and maintained.

Claim Rejections - 35 USC § 112

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

5. Claims 1-8, 10-18 and 20-28 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 1, 10, 11, 20 and 21 include limitations concerning version information in different namespaces, "when written in programming language C++...", however, the claim does not clearly establish what element of the system is written in the programming language C++, as the claim could be reasonably interpreted to indicate that multiple things could be written in C++. Consequently, the scope of the claim cannot be reasonably ascertained and the claim is thereby indefinite. As such, because the limitation regarding version information being defined in different namespaces only occurs in relation to "something" being written in C++, that "something" being indefinite, the claim is hereby being interpreted as not requiring version information to be defined in different namespaces, since the claim is not requiring anything to be written in C++.

Claims 2-8, 12-18 and 22-28 are rejected due to their dependence on a rejected base claim.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1, 8, 10, 21 and 28 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 5,974,454 to Apfel et al., in view of U.S. Patent 5,732,275 to Kullick et al.

Regarding claim 1:

Apfel et al teach:

- an information processing system ("a system for updating a software program..." in col. 14 line 8)
- at least one terminal apparatus and at least one program execution apparatus ("the computer, a database server, and a package server..." in col. 14 line 10)
- each of said at least one terminal apparatus comprises a message transmitting unit which transmits a message containing version information indicating a program version ("sending an upgrade package message from the database server to the computer..." in col. 12 lines 29-30)
- a message receiving unit which receives a message containing version information indicating a program version, from one of said at least one terminal apparatus ("sending an upgrade

package message from the database server to the computer..." in col. 12 lines 29-30. There must be a receiving unit for the computer to receive the upgrade package message.)

- a program storing unit which stores one or more program components ("the personal computer further includes a hard disk drive..." in col. 4 lines 22-23, and further, "A number of program modules may be stored in the drives..." in col. 4 line 40)
- a pre-transfer information management table which holds information on said one or more program components stored in said program storing unit ("a database query is sent from the computer...the database query...can include information such as the version of the program module component...the platform that the program module component is running on, and the languages of the program module component." in col. 2 lines 28-35. This information contained in the query must have been stored in an information table on the program storing unit.)
- a program memory unit which is allocated to an activated process, and temporarily stores at least one program component transferred from said program storing unit ("program modules may be stored in the...RAM..." in col. 4 lines 40-41)
- a post-transfer information management table which holds information on said at least one program component stored in said program memory unit ("the upgrade package is installed on computer..." in col. 11 line 1. Further, since the newly installed program must have version information, this information must overwrite the prior information that was stored in the information table for the old version of the program.
- a program executing unit which dynamically links one of said one or more program components corresponding to said version information contained in said message received by said message receiving unit, to said program memory unit, so as to enable execution of

said one of said one or more program components in said process ("the upgrade package is installed on computer..." in col. 11 line 1. The installation program must link the program to a memory unit for the program to be able to operate, and further, this linking occurs dynamically while the program installation is executing.)

substantially as claimed. Apfel et al. do not explicitly disclose changing, without stopping transaction processing, a plurality of versions of a plurality of software components which are distributed over a plurality of terminal apparatuses when more than one version of a software component may coexist. Kullick et al. disclose in an analogous distributed software updating system the ability to update multiple versions of software components without stopping transaction processing ("multiple resident version of a software program..." in col. 2 lines 31-32. Further, "the upgrading of programs stored in the memories of the client computers is carried out automatically in a manner that is transparent to the users of the computers and without interruption to the normal operation of the programs" in col. 3 lines 59-62). It would have been obvious to one of ordinary skill in the art at the time the invention was made to utilize the non-disruptive upgrading ability of Kullick et al. with the software upgrade system disclosed by Apfel et al., as this would allow the upgrading of system software while sustaining user productivity, as the system would not experience downtime due to the upgrade.

Regarding claim 8:

The rejection of claim 1 is incorporated, and further, Apfel et al disclose a management apparatus which sets and manages the one or more program components ("Auto-autoupdate is a cooperative scheduled client 'pull' routine designed to automatically update the...program module..." in col. 3 lines 39-41. The updating client is a management apparatus for the updating of the components.)

Regarding claim 10:

Apfel et al teach:

- a program execution apparatus ("The personal computer..." in col. 4 line 60)
- a message receiving unit which receives a message containing version information indicating a program version ("sending an upgrade package message from the database server to the computer..." in col. 12 lines 29-30. There must be a receiving unit for the computer to receive the upgrade package message.)
- a program storing unit which stores one or more program components ("the personal computer further includes a hard disk drive..." in col. 4 lines 22-23, and further, "A number of program modules may be stored in the drives..." in col. 4 line 40)
- a pre-transfer information management table which holds information on said one or more program components stored in said program storing unit ("a database query is sent from the computer...the database query...can include information such as the version of the program module component...the platform that the program module component is running on, and the languages of the program module component." in col. 2 lines 28-35. This information contained in the query must have been stored in an information table on the program storing unit.)
- a program memory unit which is allocated to an activated process, and temporarily stores at least one program component transferred from said program storing unit ("program modules may be stored in the...RAM..." in col. 4 lines 40-41)
- a post-transfer information management table which holds information on said at least one program component stored in said program memory unit ("the upgrade package is installed

on computer..." in col. 11 line 1. Further, since the newly installed program must have version information, this information must overwrite the prior information that was stored in the information table for the old version of the program.

- a program executing unit which dynamically links one of said one or more program components corresponding to said version information contained in said message received by said message receiving unit, to said program memory unit, so as to enable execution of said one of said one or more program components in said process ("the upgrade package is installed on computer..." in col. 11 line 1. The installation program must link the program to a memory unit for the program to be able to operate, and further, this linking occurs dynamically while the program installation is executing.)

substantially as claimed. Apfel et al. do not explicitly disclose changing, without stopping transaction processing, a plurality of versions of a plurality of software components which are distributed over a plurality of terminal apparatuses when more than one version of a software component may coexist. Kullick et al. disclose in an analogous distributed software updating system the ability to update multiple versions of software components without stopping transaction processing ("multiple resident version of a software program..." in col. 2 lines 31-32. Further, "the upgrading of programs stored in the memories of the client computers is carried out automatically in a manner that is transparent to the users of the computers and without interruption to the normal operation of the programs" in col. 3 lines 59-62). It would have been obvious to one of ordinary skill in the art at the time the invention was made to utilize the non-disruptive upgrading ability of Kullick et al. with the software upgrade system disclosed by Apfel et al., as this would allow the upgrading of system software while sustaining user productivity, as the system would not experience downtime due to the upgrade.

Regarding claim 21:

Apfel et al teach:

- a method for updating a program component ("method for updating software program..." in col. 2 lines 13-14)
- loaded in a server in an N-tier client-server environment ("the computer, a database server, and a package server..." in col. 14 line 10)
- storing in said server one or more program components ("new program module components may also be obtained from the package server..." in col. 5 lines 25-26)
- holding information on said one or more program components stored in said server, in a pre-transfer information management table ("the database server needs to determine if an upgrade is available..." in col. 6 lines 50-51. For the server to determine if an upgrade is available, it must have a table of current versions presently stored on the server.)
- transmitting a first message containing version information indicating a program version from a client to a server ("a database query is sent from the computer over the Internet to a database server...the database query...can include information such as the version of the program module component..." in col. 2 lines 28-33)
- receiving said first message by said server ("In response to receiving the database query, the database server determines..." in col. 2 lines 36-37)
- dynamically linking one of said one or more program components corresponding to said version information contained in said first message to a memory which is allocated to an activated process in said server, so as to enable execution of said one or said one or more program components in said process ("in a networked environment, program modules

depicted relative to the personal computer, or portions thereof, may be stored in the remote memory storage device." in col. 5 lines 13-16. To use the remote program module, the component would have to be linked to memory)

- holding in a post-transfer information management table information on at least one program component stored in said memory ("the servers may also provide a 'next check' date if there is not an upgrade currently available." in col. 7 lines 25-27. The date information must be stored in a table somewhere on the system for each component in memory.)
- transmitting to another server a second message containing said version information ("After the computer receives the response including the URL of the upgrade package, the computer will send a query to the package server..." in col. 7 lines 4-6)

substantially as claimed. Apfel et al. do not explicitly disclose changing, without stopping transaction processing, a plurality of versions of a plurality of software components which are distributed over a plurality of terminal apparatuses when more than one version of a software component may coexist. Kullick et al. disclose in an analogous distributed software updating system the ability to update multiple versions of software components without stopping transaction processing ("multiple resident version of a software program..." in col. 2 lines 31-32. Further, "the upgrading of programs stored in the memories of the client computers is carried out automatically in a manner that is transparent to the users of the computers and without interruption to the normal operation of the programs" in col. 3 lines 59-62). It would have been obvious to one of ordinary skill in the art at the time the invention was made to utilize the non-disruptive upgrading ability of Kullick et al. with the software upgrade system disclosed by Apfel et al., as this would allow the upgrading of system software while sustaining user productivity, as the system would not experience downtime due to the upgrade.

Regarding claim 28:

The rejection of claim 21 is incorporated, and further, Apfel et al disclose a management apparatus which sets and manages the one or more program components ("Auto-autoupdate is a cooperative scheduled client 'pull' routine designed to automatically update the...program module..." in col. 3 lines 39-41. The updating client is a management apparatus for the updating of the components.)

8. **Claims 11, 18 and 20 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent 6,493,768 to Boutcher, in view of U.S. Patent 5,732,275 to Kullick et al.**

Regarding claim 11:

Boutcher teaches:

- a distributed processing system ("a distributed computer system..." in col. 2 lines 62-63)
- including at least one client apparatus and a plurality of server apparatuses, and realizing an N-tier client-server environment (Note Figure 1 and the corresponding section of the disclosure)
- a client stub processing unit which executes stub processing of a first message which contains version information indicating a program version ("a version map, which is preferably resident in the client stub..." in col. 8 lines 35-36, and further, "the version of the server is compared against the different versions...for the client." in col. 11 lines 20-23)
- a server skeleton processing unit which executes skeleton processing of a first message containing version information indicating a program version ("determines whether version mapping is required, generally by comparing the version of the server with the version of the

client..." in col. 10 lines 54-57. A server skeleton would exist in the system for performing the actions of "marshalling, unmarshalling, sending and receiving data across a network in a distributed computer system" which Boutcher states is "generally understood in the art." in col. 11 lines 8-11)

- a distributed-object storing repository which stores one or more distributed objects ("a plurality of mass storage devices..." in col. 4 line 54)
- a pre-transfer information management table which holds information on said one or more distributed objects stored in said distributed-object storing repository ("the supported versions of the server are compared with the version of the client..." in col. 13 lines 15-16. There must be a table of version information in the server if a comparison is to be made.)
- a memory which is allocated to an activated process, and temporarily stores at least one distributed object transferred from said distributed-object storing repository ("memory devices such as RAMs...the various applications...may be transferred or downloaded to a computer system...typically by first establishing a connection between the computer system and a server-type computer..." in col. 5 lines 34-46)
- a post-transfer information management table which holds information on said at least one distributed object stored in said memory ("the supported versions of the server are compared with the version of the client..." in col. 13 lines 15-16. There must be a table of version information in the server if a comparison is to be made, and further, once the update occurs, the table must be updated to reflect the current versions.)
- a distributed object execution control unit which dynamically links one of said one or more distributed objects corresponding to said version information contained in said first message of which skeleton processing is executed by said server skeleton processing unit, to said

memory, so as to enable execution of a least one function in said one of said one or more distributed objects ("responds to the mapping by requesting...the second computer process to execute the second version of the one remote procedure." in col. 15 lines 2-4. For the computer to execute the procedure, it must have been dynamically linked to a memory space)

- a server stub processing unit which executes stub processing of a second message containing said version information so as to transmit the second message to another of said plurality of server apparatuses (Note Figure 2B, item 44 and the corresponding section of the disclosure.)

substantially as claimed. While Boutcher discloses the ability to have a plurality of versions of software components, Boutcher does not explicitly disclose changing, without stopping transaction processing, a plurality of versions of a plurality of software components which are distributed over a plurality of terminal apparatuses when more than one version of a software component may coexist. Kullick et al. disclose in an analogous distributed software updating system the ability to update multiple versions of software components without stopping transaction processing ("multiple resident version of a software program..." in col. 2 lines 31-32. Further, "the upgrading of programs stored in the memories of the client computers is carried out automatically in a manner that is transparent to the users of the computers and without interruption to the normal operation of the programs" in col. 3 lines 59-62). It would have been obvious to one of ordinary skill in the art at the time the invention was made to utilize the non-disruptive upgrading ability of Kullick et al. with the software upgrade system disclosed by Boutcher, as this would allow the upgrading of system software while sustaining user productivity, as the system would not experience downtime due to the upgrade.

Regarding claim 18:

The rejection of claim 11 is incorporated, and further, Boutcher discloses a management server which sets and manages one or more distributed objects stored in a distributed-object storing repository and the information on one or more distributed objects held in a pre-transfer information management table ("a server-type computer...which downloads or transfers a program product to other computers systems..." in col. 5 lines 49-51)

Regarding claim 20:

Boutcher teaches:

- a server apparatus ("server systems..." in col. 4 line 59)
- a server skeleton processing unit which executes skeleton processing of a first message containing version information indicating a program version ("determines whether version mapping is required, generally by comparing the version of the server with the version of the client..." in col. 10 lines 54-57. A server skeleton would exist in the system for performing the actions of "marshalling, unmarshalling, sending and receiving data across a network in a distributed computer system" which Boutcher states is "generally understood in the art." in col. 11 lines 8-11)
- a distributed-object storing repository which stores one or more distributed objects ("a plurality of mass storage devices..." in col. 4 line 54)
- a pre-transfer information management table which holds information on said one or more distributed objects stored in said distributed-object storing repository ("the supported

versions of the server are compared with the version of the client..." in col. 13 lines 15-16. There must be a table of version information in the server if a comparison is to be made.)

- a memory which is allocated to an activated process, and temporarily stores at least one distributed object transferred from said distributed-object storing repository ("memory devices such as RAMs...the various applications...may be transferred or downloaded to a computer system...typically by first establishing a connection between the computer system and a server-type computer..." in col. 5 lines 34-46)
- a post-transfer information management table which holds information on said at least one distributed object stored in said memory ("the supported versions of the server are compared with the version of the client..." in col. 13 lines 15-16. There must be a table of version information in the server if a comparison is to be made, and further, once the update occurs, the table must be updated to reflect the current versions.)
- a distributed object execution control unit which dynamically links one of said one or more distributed objects corresponding to said version information contained in said first message of which skeleton processing is executed by said server skeleton processing unit, to said memory, so as to enable execution of a least one function in said one of said one or more distributed objects ("responds to the mapping by requesting...the second computer process to execute the second version of the one remote procedure." in col. 15 lines 2-4. For the computer to execute the procedure, it must have been linked to a memory space.)
- a server stub processing unit which executes stub processing of a second message containing said version information so as to transmit the second message to another of said plurality of server apparatuses (Note Figure 2B, item 44 and the corresponding section of the disclosure.)

substantially as claimed. While Boutcher discloses the ability to have a plurality of versions of software components, Boutcher does not explicitly disclose changing, without stopping transaction processing, a plurality of versions of a plurality of software components which are distributed over a plurality of terminal apparatuses when more than one version of a software component may coexist. Kullick et al. disclose in an analogous distributed software updating system the ability to update multiple versions of software components without stopping transaction processing ("multiple resident version of a software program..." in col. 2 lines 31-32. Further, "the upgrading of programs stored in the memories of the client computers is carried out automatically in a manner that is transparent to the users of the computers and without interruption to the normal operation of the programs" in col. 3 lines 59-62). It would have been obvious to one of ordinary skill in the art at the time the invention was made to utilize the non-disruptive upgrading ability of Kullick et al. with the software upgrade system disclosed by Boutcher, as this would allow the upgrading of system software while sustaining user productivity, as the system would not experience downtime due to the upgrade.

9. Claims 2, 3, 6, 7, 22, 23, 26 and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 5,974,454 to Apfel et al in view of U.S. Patent 5,732,275 to Kullick et al., further in view of U.S. Patent 6,381,735 to Hunt.

Regarding claim 2:

The rejection of claim 1 is incorporated, and further, neither Apfel et al. nor Kullick et al. disclose a reference count which indicates the number of executions in which said each of said at least one program components are currently referring to. Hunt discloses in an analogous memory

management system a reference count that indicates the number of executions in which said each of said at least one program components are currently referring to ("a component knows exactly how many clients have references to it...its reference count..." in col. 13 lines 34-35) It would have been obvious to someone of ordinary skill in the art at the time the invention was made to use the reference counter of Hunt with the component upgrading system of Apfel et al. modified by Kullick et al., as this would be used to ensure that the program component in the system disclosed by Apfel et al. modified by Kullick et al. is not updated while the component is still in use.

Regarding claim 3:

The rejection of claim 2 is incorporated, and further, neither Apfel et al. nor Kullick et al. disclose removing one of said at least one program component from said program memory unit when a reference count for the program component is zero. Hunt discloses in an analogous memory management system removing a program component from a program memory unit when a reference count for the program component is zero ("When its reference count goes to zero, the component is responsible for freeing itself from memory." in col. 13 lines 35-37) It would have been obvious to someone of ordinary skill in the art at the time the invention was made to use the reference counting system of Hunt with the component upgrading system of Apfel et al. modified by Kullick et al., as this would recover memory space when it is no longer being used, thereby offering more memory to alternate components in the system disclosed by Apfel et al. modified by Kullick et al.

Regarding claim 6:

The rejection of claim 1 is incorporated, and further, note the rejection regarding claim 2. The evaluation count is identical to a reference count, as the reference counter is incremented when the system is evaluating the component the reference points to.

Regarding claim 7:

The rejection of claim 6 is incorporated, and further, note the rejection regarding claim 3.

Regarding claim 22:

The rejection of claim 21 is incorporated, and further, neither Apfel et al. nor Kullick et al. disclose a reference count which indicates the number of executions in which said each of said at least one program components are currently referring to. Hunt discloses in an analogous memory management system a reference count that indicates the number of executions in which said each of said at least one program components are currently referring to ("a component knows exactly how many clients have references to it...its reference count..." in col. 13 lines 34-35) It would have been obvious to someone of ordinary skill in the art at the time the invention was made to use the reference counter of Hunt with the component upgrading system of Apfel et al. modified by Kullick et al., as this would be used to ensure that the program component in the system disclosed by Apfel et al. modified by Kullick et al. is not updated while the component is still in use.

Regarding claim 23:

The rejection of claim 22 is incorporated, and further, neither Apfel et al. nor Kullick et al. disclose removing one of said at least one program component from said program memory unit when a reference count for the program component is zero. Hunt discloses in an analogous memory

management system removing a program component from a program memory unit when a reference count for the program component is zero ("When its reference count goes to zero, the component is responsible for freeing itself from memory." in col. 13 lines 35-37) It would have been obvious to someone of ordinary skill in the art at the time the invention was made to use the reference counting system of Hunt with the component upgrading system of Apfel et al. modified by Kullick et al., as this would recover memory space when it is no longer being used, thereby offering more memory to alternate components in the system disclosed by Apfel et al. modified by Kullick et al.

Regarding claim 26:

The rejection of claim 21 is incorporated, and further, note the rejection regarding claim 22. The evaluation count is identical to a reference count, as the reference counter is incremented when the system is evaluating the component the reference points to.

Regarding claim 27:

The rejection of claim 26 is incorporated, and further, note the rejection regarding claim 23.

10. Claims 4, 5, 24 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 5,974,454 to Apfel et al in view of U.S. Patent 5,732,275 to Kullick et al., further in view of U.S. Patent 5,940,827 to Hapner et al.

Regarding claim 4:

The rejection of claim 1 is incorporated, and further, neither Apfel et al. nor Kullick et al. disclose an evaluation flag which indicates whether or not a program component is under evaluation. Hapner et al discloses in an analogous distributed computing system an evaluation flag which indicates whether or not a program component is under evaluation ("having a data structure which includes a true or false flag, thereby preventing all other threads from operating on this condition variable." in col. 10 lines 28-30). It would have been obvious to someone of ordinary skill in the art at the time the invention was made to use the thread locking system of Hapner et al with the component upgrading system of Apfel et al. modified by Kullick et al., as this would ensure that steps of the system occur in a desired order, as suggested by Hapner et al in col. 10 lines 21-22.

Regarding claim 5:

The rejection of claim 4 is incorporated, and further, neither Apfel et al. nor Kullick et al. disclose a program executing unit executing a program component for which said evaluation flag is contained in said message, in a separate process, when the evaluation flag indicates that said program component for which said evaluation flag is contained in said message is under evaluation. Hapner et al discloses an evaluation flag as claimed ("The first thread may then evaluate the true or false flag, performing the desired action if the flag value is true." in col. 10 lines 32-33.). It would have been obvious to someone of ordinary skill in the art at the time the invention was made to use the thread locking system of Hapner et al with the component upgrading system of Apfel et al. modified by Kullick et al., as this would ensure that steps of the system occur in a desired order, as suggested by Hapner et al in col. 10 lines 21-22.

Regarding claim 24:

The rejection of claim 21 is incorporated, and further, note the rejection regarding claim 4.

Regarding claim 25:

The rejection of claim 24 is incorporated, and further, note the rejection regarding claim 5.

11. Claims 12, 13, 16 and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 6,493,768 to Boutcher in view of U.S. Patent 5,732,275 to Kullick et al., further in view of U.S. Patent 6,381,735 to Hunt.

Regarding claim 12:

The rejection of claim 11 is incorporated, and further, neither Boutcher nor Kullick et al. disclose a reference count which indicates the number of executions in which said each of said at least one program components are currently referring to. Hunt discloses in an analogous distributed system a reference count that indicates the number of executions in which said each of said at least one program components are currently referring to ("a component knows exactly how many clients have references to it...its reference count..." in col. 13 lines 34-35) It would have been obvious to someone of ordinary skill in the art at the time the invention was made to use the reference counter of Hunt with the version management system of Boutcher modified by Kullick et al., as this would be used to ensure that the program component in the system disclosed by Boutcher modified by Kullick et al. is not updated or remapped while the component is still in use.

Regarding claim 13:

The rejection of claim 12 is incorporated, and further, neither Boutcher nor Kullick et al. disclose removing one of said at least one program component from said program memory unit when a reference count for the program component is zero. Hunt discloses in an analogous distributed system removing a program component from a program memory unit when a reference count for the program component is zero ("When its reference count goes to zero, the component is responsible for freeing itself from memory." in col. 13 lines 35-37) It would have been obvious to someone of ordinary skill in the art at the time the invention was made to use the reference counting system of Hunt with the version management system of Boutcher modified by Kullick et al., as this would recover memory space when it is no longer being used, thereby offering more memory to alternate processes in the system disclosed by Boutcher modified by Kullick et al.

Regarding claim 16:

The rejection of claim 11 is incorporated, and further, note the rejection regarding claim 12. The evaluation count is identical to a reference count, as the reference counter is incremented when the system is evaluating the component the reference points to.

Regarding claim 17:

The rejection of claim 16 is incorporated, and further, note the rejection of claim 13.

12. Claims 14 and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 6,493,768 to Boutcher in view of U.S. Patent 5,732,275 to Kullick et al., further in view of U.S. Patent 5,940,827 to Hapner et al.

Regarding claim 14:

The rejection of claim 11 is incorporated, and further, neither Boutcher nor Kullick et al. disclose an evaluation flag which indicates whether or not a program component is under evaluation. Hapner et al discloses in an analogous distributed computing system an evaluation flag which indicates whether or not a program component is under evaluation ("having a data structure which includes a true or false flag, thereby preventing all other threads from operating on this condition variable." in col. 10 lines 28-30). It would have been obvious to someone of ordinary skill in the art at the time the invention was made to use the thread locking system of Hapner et al with the version management system of Boutcher modified by Kullick et al., as this would ensure that steps of the system occur in a desired order, as suggested by Hapner et al in col. 10 lines 21-22.

Regarding claim 15:

The rejection of claim 14 is incorporated, and further, neither Boutcher nor Kullick et al. disclose a program executing unit executing a program component for which said evaluation flag is contained in said message, in a separate process, when the evaluation flag indicates that said program component for which said evaluation flag is contained in said message is under evaluation. Hapner et al discloses an evaluation flag as claimed ("The first thread may then evaluate the true or false flag, performing the desired action if the flag value is true." in col. 10 lines 32-33.). It would have been obvious to someone of ordinary skill in the art at the time the invention was made to use the thread locking system of Hapner et al with the version management system of Boutcher modified by Kullick et al., as this would ensure that steps of the system occur in a desired order, as suggested by Hapner et al in col. 10 lines 21-22.

Conclusion

13. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action."

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Trenton J. Roche whose telephone number is (571) 272-3733. The examiner can normally be reached on Monday - Friday, 9:00 am - 5:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571) 272-3719. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Trenton J Roche
Examiner
Art Unit 2193

TJR

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100